

In [44]:

```
# Import necessary Libraries  
import pandas as pd  
import seaborn as sns  
import numpy as np  
import matplotlib.pyplot as plt  
from matplotlib import dates  
from datetime import datetime
```

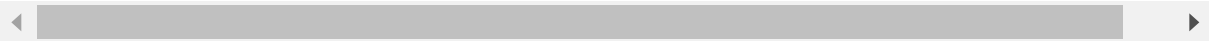
In [45]:

```
# Load dataset
data = pd.read_csv('Walmart_Store_sales.csv')
data
```

Out[45]:

| | Store | Date | Weekly_Sales | Holiday_Flag | Temperature | Fuel_Price | CPI | Unemployr |
|------|-------|------------|--------------|--------------|-------------|------------|------------|-----------|
| 0 | 1 | 05-02-2010 | 1643690.90 | 0 | 42.31 | 2.572 | 211.096358 | 8. |
| 1 | 1 | 12-02-2010 | 1641957.44 | 1 | 38.51 | 2.548 | 211.242170 | 8. |
| 2 | 1 | 19-02-2010 | 1611968.17 | 0 | 39.93 | 2.514 | 211.289143 | 8. |
| 3 | 1 | 26-02-2010 | 1409727.59 | 0 | 46.63 | 2.561 | 211.319643 | 8. |
| 4 | 1 | 05-03-2010 | 1554806.68 | 0 | 46.50 | 2.625 | 211.350143 | 8. |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 6430 | 45 | 28-09-2012 | 713173.95 | 0 | 64.88 | 3.997 | 192.013558 | 8. |
| 6431 | 45 | 05-10-2012 | 733455.07 | 0 | 64.89 | 3.985 | 192.170412 | 8. |
| 6432 | 45 | 12-10-2012 | 734464.36 | 0 | 54.47 | 4.000 | 192.327265 | 8. |
| 6433 | 45 | 19-10-2012 | 718125.53 | 0 | 56.47 | 3.969 | 192.330854 | 8. |
| 6434 | 45 | 26-10-2012 | 760281.43 | 0 | 58.85 | 3.882 | 192.308899 | 8. |

6435 rows × 8 columns



Data Preparation

In [46]:

```
# Convert date to datetime format and show dataset information
data['Date'] = pd.to_datetime(data['Date'])
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  -
 0   Store           6435 non-null   int64
 1   Date            6435 non-null   datetime64[ns]
 2   Weekly_Sales    6435 non-null   float64
 3   Holiday_Flag    6435 non-null   int64
 4   Temperature     6435 non-null   float64
 5   Fuel_Price      6435 non-null   float64
 6   CPI             6435 non-null   float64
 7   Unemployment    6435 non-null   float64
dtypes: datetime64[ns](1), float64(5), int64(2)
memory usage: 402.3 KB
```

In [47]:

```
# checking for missing values
data.isnull().sum()
```

Out[47]:

```
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
dtype: int64
```

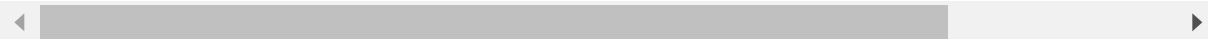
In [48]:

```
# Splitting Date and create new columns (Day, Month, and Year)
data["Day"] = pd.DatetimeIndex(data['Date']).day
data['Month'] = pd.DatetimeIndex(data['Date']).month
data['Year'] = pd.DatetimeIndex(data['Date']).year
data
```

Out[48]:

| | Store | Date | Weekly_Sales | Holiday_Flag | Temperature | Fuel_Price | CPI | Unemployr |
|------|-------|------------|--------------|--------------|-------------|------------|------------|-----------|
| 0 | 1 | 2010-05-02 | 1643690.90 | 0 | 42.31 | 2.572 | 211.096358 | 8 |
| 1 | 1 | 2010-12-02 | 1641957.44 | 1 | 38.51 | 2.548 | 211.242170 | 8 |
| 2 | 1 | 2010-02-19 | 1611968.17 | 0 | 39.93 | 2.514 | 211.289143 | 8 |
| 3 | 1 | 2010-02-26 | 1409727.59 | 0 | 46.63 | 2.561 | 211.319643 | 8 |
| 4 | 1 | 2010-05-03 | 1554806.68 | 0 | 46.50 | 2.625 | 211.350143 | 8 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 6430 | 45 | 2012-09-28 | 713173.95 | 0 | 64.88 | 3.997 | 192.013558 | 8 |
| 6431 | 45 | 2012-05-10 | 733455.07 | 0 | 64.89 | 3.985 | 192.170412 | 8 |
| 6432 | 45 | 2012-12-10 | 734464.36 | 0 | 54.47 | 4.000 | 192.327265 | 8 |
| 6433 | 45 | 2012-10-19 | 718125.53 | 0 | 56.47 | 3.969 | 192.330854 | 8 |
| 6434 | 45 | 2012-10-26 | 760281.43 | 0 | 58.85 | 3.882 | 192.308899 | 8 |

6435 rows × 11 columns



Q1: Which store has minimum and maximum sales?

In [49]:

```
plt.figure(figsize=(15,7))

# Sum Weekly_Sales for each store, then sortded by total sales
total_sales_for_each_store = data.groupby('Store')['Weekly_Sales'].sum().sort_values()
total_sales_for_each_store_array = np.array(total_sales_for_each_store) # convert to array

# Assigning a specific color for the stores have the lowest and highest sales
clrs = ['lightsteelblue' if ((x < max(total_sales_for_each_store_array)) and (x > min(total_sales_for_each_store_array))) else 'midnightblue' for x in total_sales_for_each_store_array]

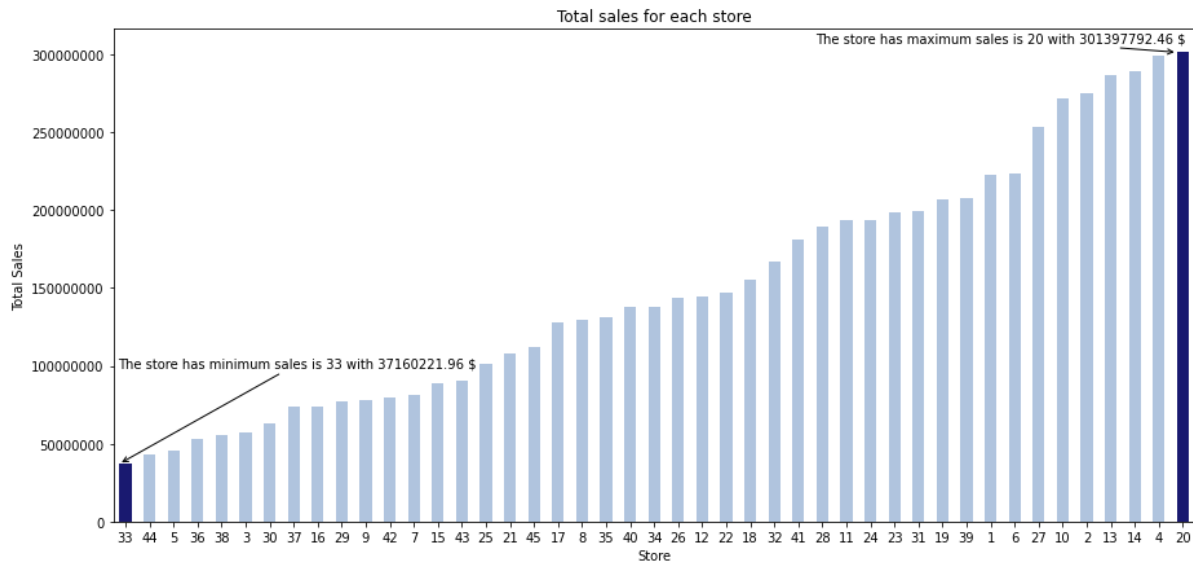
ax = total_sales_for_each_store.plot(kind='bar',color=clrs);

# store have minimum sales
p = ax.patches[0]
print(type(p.get_height()))
ax.annotate("The store has minimum sales is 33 with {0:.2f} $".format((p.get_height()))), x
y=(p.get_x(), p.get_height()), xycoords='data',
      xytext=(0.17, 0.32), textcoords='axes fraction',
      arrowprops=dict(arrowstyle="->", connectionstyle="arc3"),
      horizontalalignment='center', verticalalignment='center')

# store have maximum sales
p = ax.patches[44]
ax.annotate("The store has maximum sales is 20 with {0:.2f} $".format((p.get_height()))), x
y=(p.get_x(), p.get_height()), xycoords='data',
      xytext=(0.82, 0.98), textcoords='axes fraction',
      arrowprops=dict(arrowstyle="->", connectionstyle="arc3"),
      horizontalalignment='center', verticalalignment='center')

# plot properties
plt.xticks(rotation=0)
plt.ticklabel_format(useOffset=False, style='plain', axis='y')
plt.title('Total sales for each store')
plt.xlabel('Store')
plt.ylabel('Total Sales');
```

```
<class 'numpy.float64'>
```



Q2: Which store has maximum standard deviation i.e., the sales vary a lot. Also, find out the coefficient of mean to standard deviation?

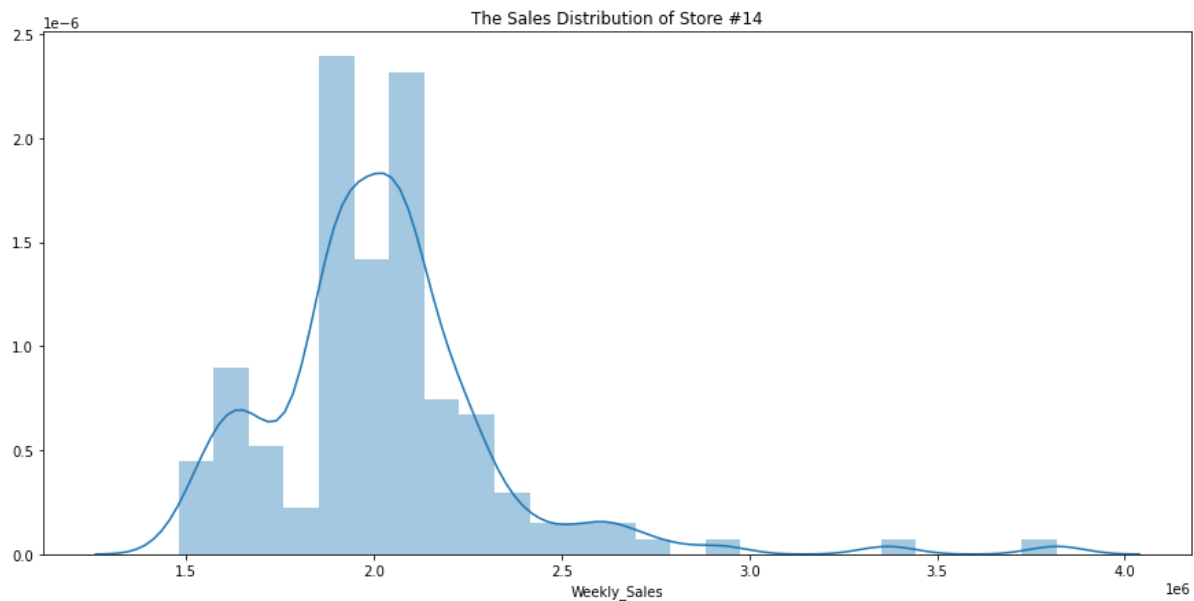
In [50]:

```
# Which store has maximum standard deviation
data_std = pd.DataFrame(data.groupby('Store')['Weekly_Sales'].std().sort_values(ascending=
False))
print("The store has maximum standard deviation is "+str(data_std.head(1).index[0])+" with
{0:.0f} $".format(data_std.head(1).Weekly_Sales[data_std.head(1).index[0]]))
```

The store has maximum standard deviation is 14 with 317570 \$

In [51]:

```
# Distribution of store has maximum standard deviation
plt.figure(figsize=(15,7))
sns.distplot(data[data['Store'] == data_std.head(1).index[0]]['Weekly_Sales'])
plt.title('The Sales Distribution of Store #' + str(data_std.head(1).index[0]));
```



In [52]:

```
# Coefficient of mean to standard deviation
coef_mean_std = pd.DataFrame(data.groupby('Store')['Weekly_Sales'].std() / data.groupby('S
tore')['Weekly_Sales'].mean())
coef_mean_std = coef_mean_std.rename(columns={'Weekly_Sales':'Coefficient of mean to stand
ard deviation'})
coef_mean_std
```


Out[52]:

| Coefficient of mean to standard deviation | |
|---|----------|
| Store | |
| 1 | 0.100292 |
| 2 | 0.123424 |
| 3 | 0.115021 |
| 4 | 0.127083 |
| 5 | 0.118668 |
| 6 | 0.135823 |
| 7 | 0.197305 |
| 8 | 0.116953 |
| 9 | 0.126895 |
| 10 | 0.159133 |
| 11 | 0.122262 |
| 12 | 0.137925 |
| 13 | 0.132514 |
| 14 | 0.157137 |
| 15 | 0.193384 |
| 16 | 0.165181 |
| 17 | 0.125521 |
| 18 | 0.162845 |
| 19 | 0.132680 |
| 20 | 0.130903 |
| 21 | 0.170292 |
| 22 | 0.156783 |
| 23 | 0.179721 |
| 24 | 0.123637 |
| 25 | 0.159860 |
| 26 | 0.110111 |
| 27 | 0.135155 |
| 28 | 0.137330 |
| 29 | 0.183742 |
| 30 | 0.052008 |
| 31 | 0.090161 |
| 32 | 0.118310 |
| 33 | 0.092868 |

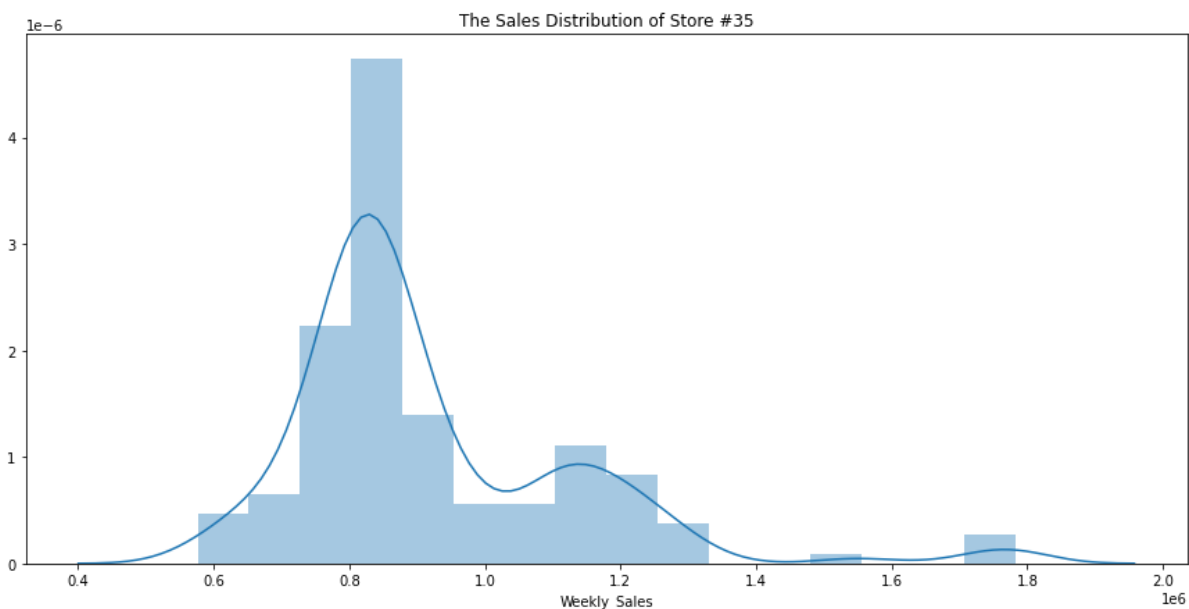
Coefficient of mean to standard deviation

Store

| | |
|----|----------|
| 34 | 0.108225 |
| 35 | 0.229681 |
| 36 | 0.162579 |
| 37 | 0.042084 |
| 38 | 0.110875 |
| 39 | 0.149908 |
| 40 | 0.123430 |
| 41 | 0.148177 |
| 42 | 0.090335 |
| 43 | 0.064104 |
| 44 | 0.081793 |
| 45 | 0.165613 |

In [53]:

```
# Distribution of store has maximum coefficient of mean to standard deviation
coef_mean_std_max = coef_mean_std.sort_values(by='Coefficient of mean to standard deviation')
plt.figure(figsize=(15,7))
sns.distplot(data[data['Store'] == coef_mean_std_max.tail(1).index[0]]['Weekly_Sales'])
plt.title('The Sales Distribution of Store #' + str(coef_mean_std_max.tail(1).index[0]));
```



Q3: Which store/s has good quarterly growth rate in Q3'2012

```
plt.figure(figsize=(15,7))
```

Sales for third quarterly in 2012

```
Q3 = data[(data["Date"] > '2012-07-01') & (data["Date"] < '2012-09-30')].groupby('Store')['Weekly_Sales'].sum()
```

Sales for second quarterly in 2012

```
Q2 = data[(data["Date"] > '2012-04-01') & (data["Date"] < '2012-06-30')].groupby('Store')['Weekly_Sales'].sum()
```

Plotting the difference between sales for second and third quarterly

```
Q2.plot(ax=Q3.plot('bar',legend=True),kind='bar',color='r',alpha=0.2,legend=True); plt.legend(["Q3' 2012", "Q2' 2012"]);
```

In [54]:

```
# store/s has good quarterly growth rate in Q3'2012 - .sort_values(by='Weekly_Sales')
print('Store have good quarterly growth rate in Q3'2012 is Store '+str(Q3.idxmax())+' With '+str(Q3.max())+' $')
```

```
Store have good quarterly growth rate in Q3'2012 is Store 4 With 25652119.35
$
```

Q4: Some holidays have a negative impact on sales. Find out holidays which have higher sales than the mean sales in non-holiday season for all stores together

Holiday Events:

- Super Bowl: 12-Feb-10, 11-Feb-11, 10-Feb-12, 8-Feb-13
- Labour Day: 10-Sep-10, 9-Sep-11, 7-Sep-12, 6-Sep-13
- Thanksgiving: 26-Nov-10, 25-Nov-11, 23-Nov-12, 29-Nov-13
- Christmas: 31-Dec-10, 30-Dec-11, 28-Dec-12, 27-Dec-13

In [55]:

```
def plot_line(df, holiday_dates, holiday_label):
    fig, ax = plt.subplots(figsize = (15,5))
    ax.plot(df['Date'], df['Weekly_Sales'], label=holiday_label)

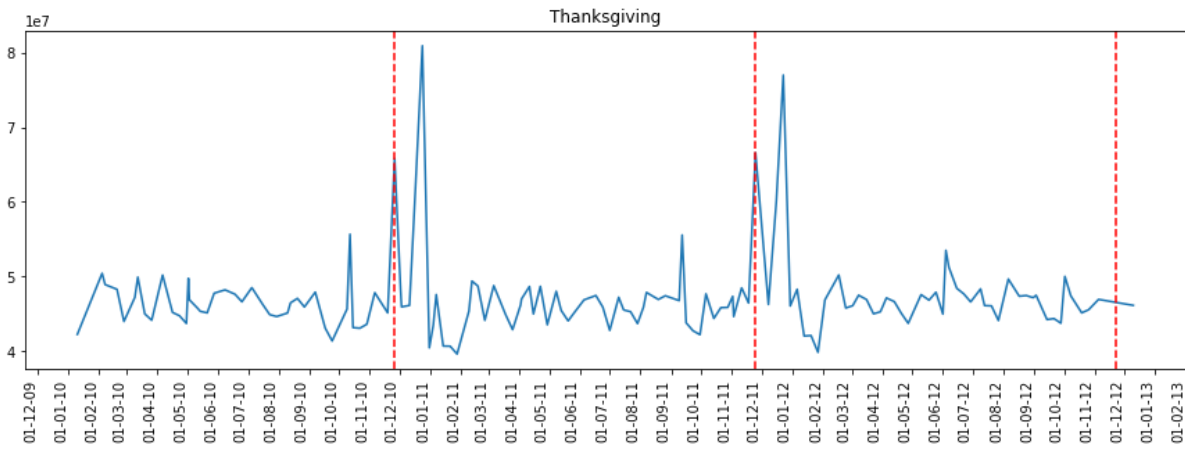
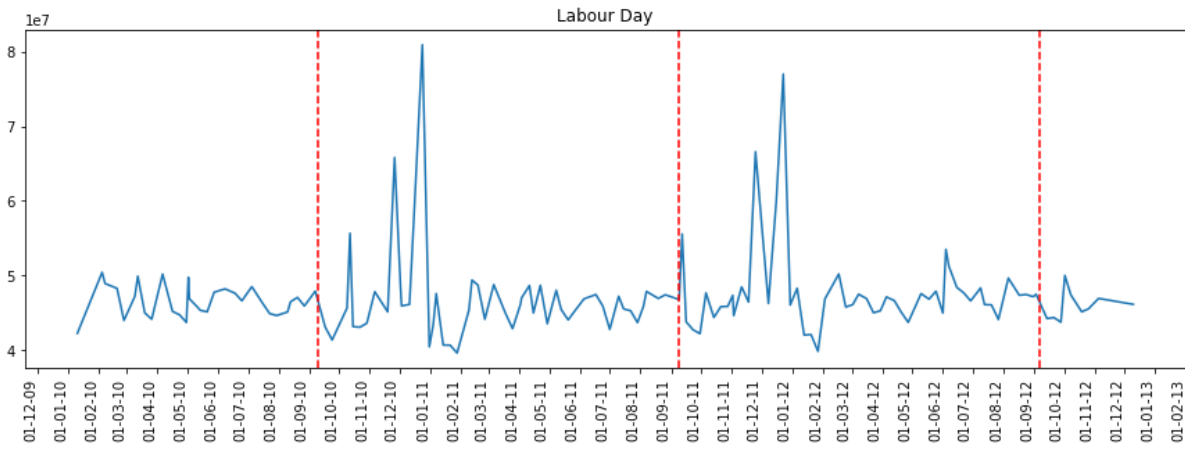
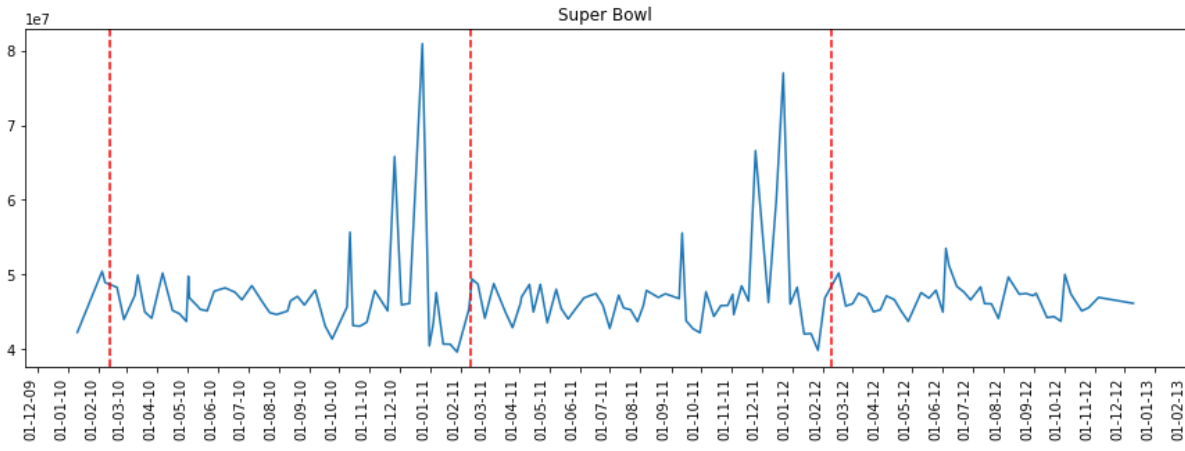
    for day in holiday_dates:
        day = datetime.strptime(day, '%d-%m-%Y')
        plt.axvline(x=day, linestyle='--', c='r')

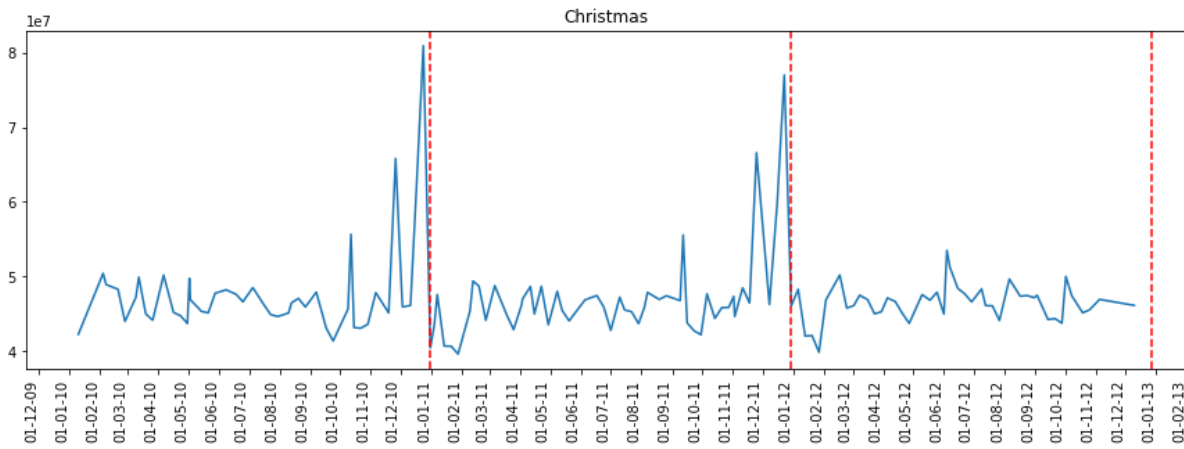
    plt.title(holiday_label)
    x_dates = df['Date'].dt.strftime('%Y-%m-%d').sort_values().unique()
    xfmt = dates.DateFormatter('%d-%m-%y')
    ax.xaxis.set_major_formatter(xfmt)
    ax.xaxis.set_major_locator(dates.DayLocator(1))
    plt.gcf().autofmt_xdate(rotation=90)
    plt.show()

total_sales = data.groupby('Date')['Weekly_Sales'].sum().reset_index()
Super_Bowl = ['12-2-2010', '11-2-2011', '10-2-2012']
Labour_Day = ['10-9-2010', '9-9-2011', '7-9-2012']
Thanksgiving = ['26-11-2010', '25-11-2011', '23-11-2012']
Christmas = ['31-12-2010', '30-12-2011', '28-12-2012']

plot_line(total_sales, Super_Bowl, 'Super Bowl')
plot_line(total_sales, Labour_Day, 'Labour Day')
plot_line(total_sales, Thanksgiving, 'Thanksgiving')
plot_line(total_sales, Christmas, 'Christmas')
```

Retail Analysis with Walmart Data





The sales increased during thanksgiving. And the sales decreased during christmas.

In [56]:

```
data.loc[data.Date.isin(Super_Bowl)]
```

Out[56]:

| | Store | Date | Weekly_Sales | Holiday_Flag | Temperature | Fuel_Price | CPI | Unemployr |
|--|-------|------------|--------------|--------------|-------------|------------|------------|-----------|
| | 1 | 2010-12-02 | 1641957.44 | 1 | 38.51 | 2.548 | 211.242170 | 8 |
| | 53 | 2011-11-02 | 1649614.93 | 1 | 36.39 | 3.022 | 212.936705 | 7 |
| | 105 | 2012-10-02 | 1802477.43 | 1 | 48.02 | 3.409 | 220.265178 | 7 |
| | 144 | 2010-12-02 | 2137809.50 | 1 | 38.49 | 2.548 | 210.897994 | 8 |
| | 196 | 2011-11-02 | 2168041.61 | 1 | 33.19 | 3.022 | 212.592862 | 8 |
| | ... | ... | ... | ... | ... | ... | ... | ... |
| | 6202 | 2011-11-02 | 307486.73 | 1 | 30.83 | 3.034 | 127.859129 | 7 |
| | 6254 | 2012-10-02 | 325377.97 | 1 | 33.73 | 3.116 | 130.384903 | 5 |
| | 6293 | 2010-12-02 | 656988.64 | 1 | 27.73 | 2.773 | 181.982317 | 8 |
| | 6345 | 2011-11-02 | 766456.00 | 1 | 30.30 | 3.239 | 183.701613 | 8 |
| | 6397 | 2012-10-02 | 803657.12 | 1 | 37.00 | 3.640 | 189.707605 | 8 |

135 rows × 11 columns



In [57]:

```
# Yearly Sales in holidays
```

```
Super_Bowl_df = pd.DataFrame(data.loc[data.Date.isin(Super_Bowl)].groupby('Year')['Weekly_Sales'].sum())
```

```
Thanksgiving_df = pd.DataFrame(data.loc[data.Date.isin(Thanksgiving)].groupby('Year')['Weekly_Sales'].sum())
```

```
Labour_Day_df = pd.DataFrame(data.loc[data.Date.isin(Labour_Day)].groupby('Year')['Weekly_Sales'].sum())
```

```
Christmas_df = pd.DataFrame(data.loc[data.Date.isin(Christmas)].groupby('Year')['Weekly_Sales'].sum())
```

```
Super_Bowl_df.plot(kind='bar', legend=False, title='Yearly Sales in Super Bowl holiday')
```

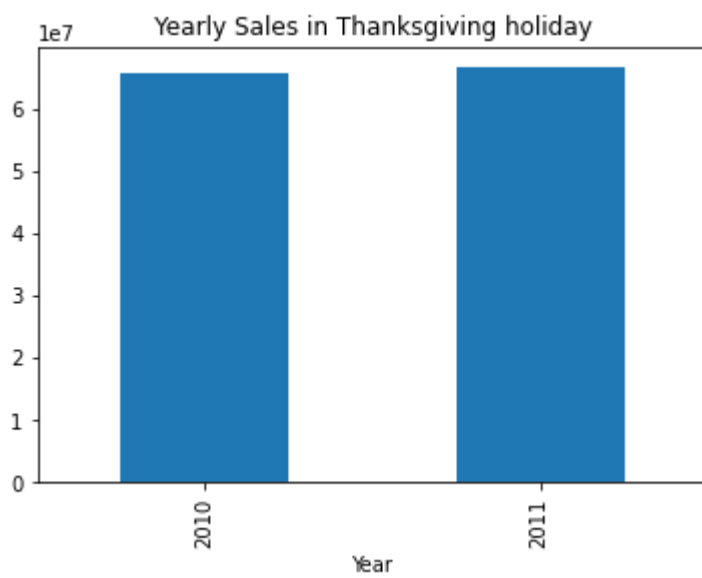
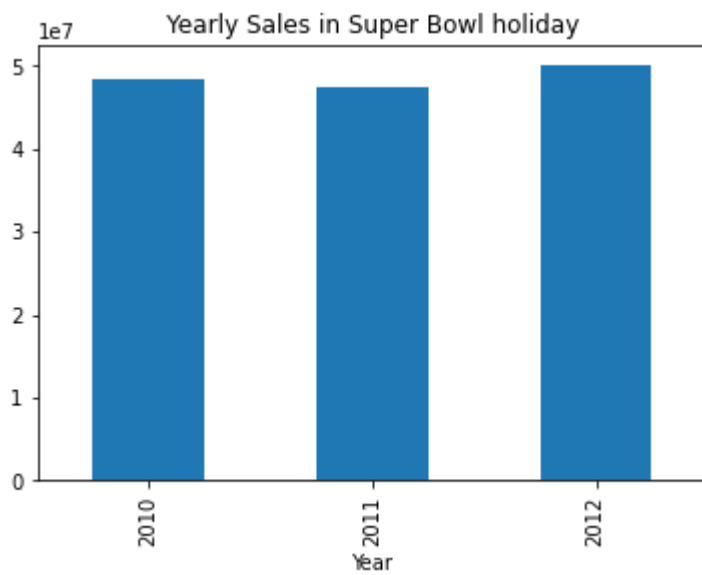
```
Thanksgiving_df.plot(kind='bar', legend=False, title='Yearly Sales in Thanksgiving holiday')
```

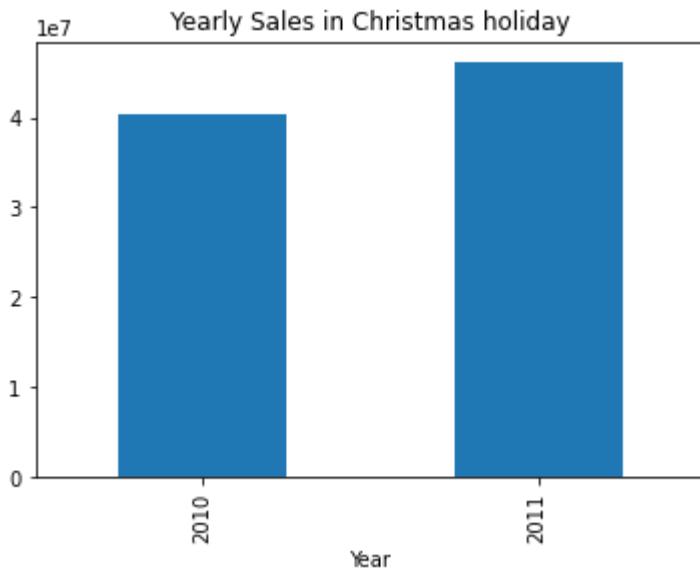
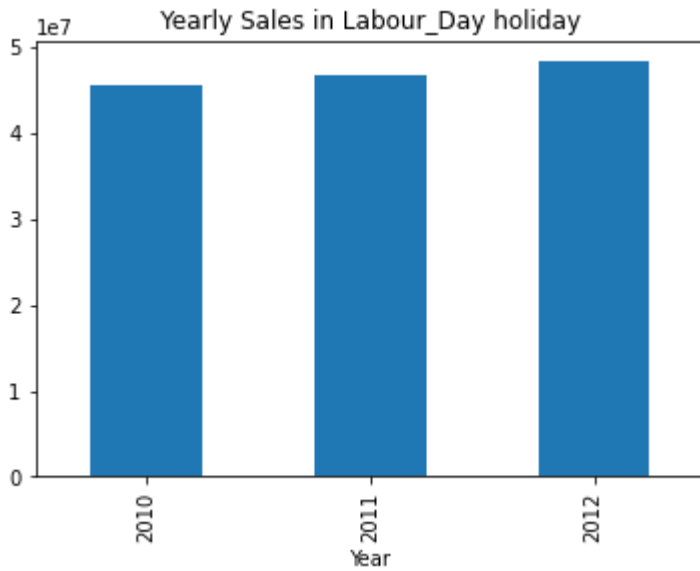
```
Labour_Day_df.plot(kind='bar', legend=False, title='Yearly Sales in Labour_Day holiday')
```

```
Christmas_df.plot(kind='bar', legend=False, title='Yearly Sales in Christmas holiday')
```

Out[57]:

```
<AxesSubplot:title={'center':'Yearly Sales in Christmas holiday'}, xlabel='Year'>
```





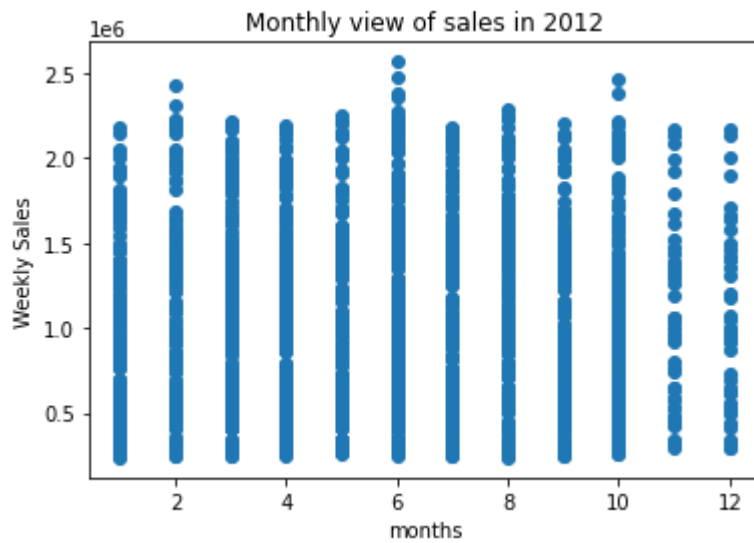
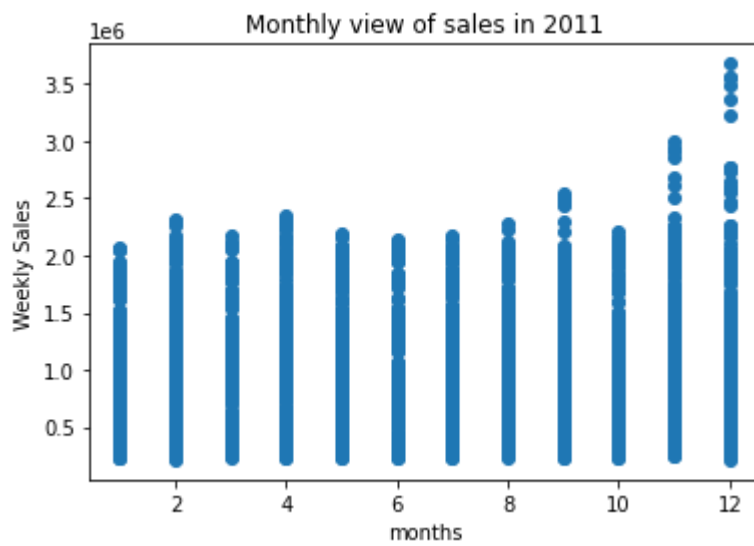
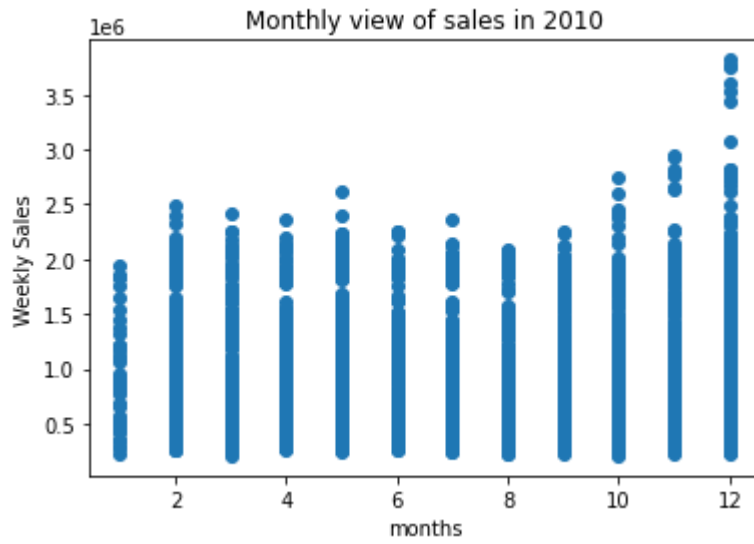
Q5: Provide a monthly and semester view of sales in units and give insights

In [58]:

```
# Monthly view of sales for each years
plt.scatter(data[data.Year==2010]["Month"],data[data.Year==2010]["Weekly_Sales"])
plt.xlabel("months")
plt.ylabel("Weekly Sales")
plt.title("Monthly view of sales in 2010")
plt.show()

plt.scatter(data[data.Year==2011]["Month"],data[data.Year==2011]["Weekly_Sales"])
plt.xlabel("months")
plt.ylabel("Weekly Sales")
plt.title("Monthly view of sales in 2011")
plt.show()

plt.scatter(data[data.Year==2012]["Month"],data[data.Year==2012]["Weekly_Sales"])
plt.xlabel("months")
plt.ylabel("Weekly Sales")
plt.title("Monthly view of sales in 2012")
plt.show()
```

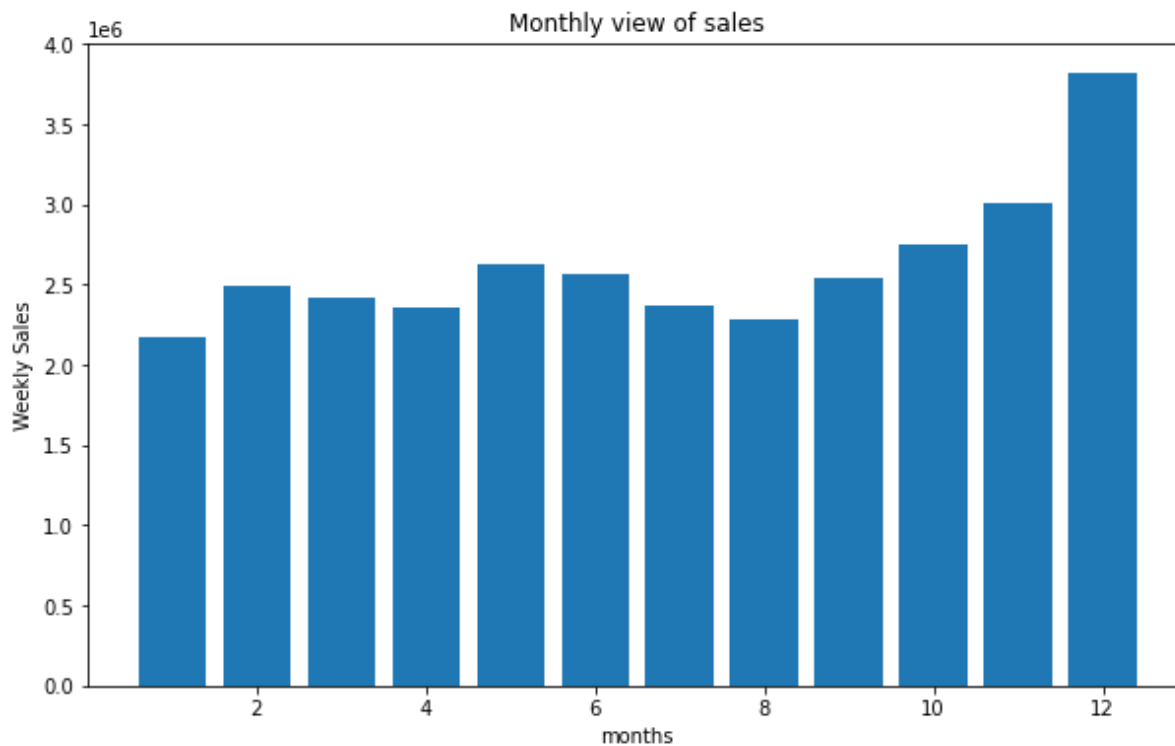


In [59]:

```
# Monthly view of sales for all years
plt.figure(figsize=(10,6))
plt.bar(data["Month"],data["Weekly_Sales"])
plt.xlabel("months")
plt.ylabel("Weekly Sales")
plt.title("Monthly view of sales")
```

Out[59]:

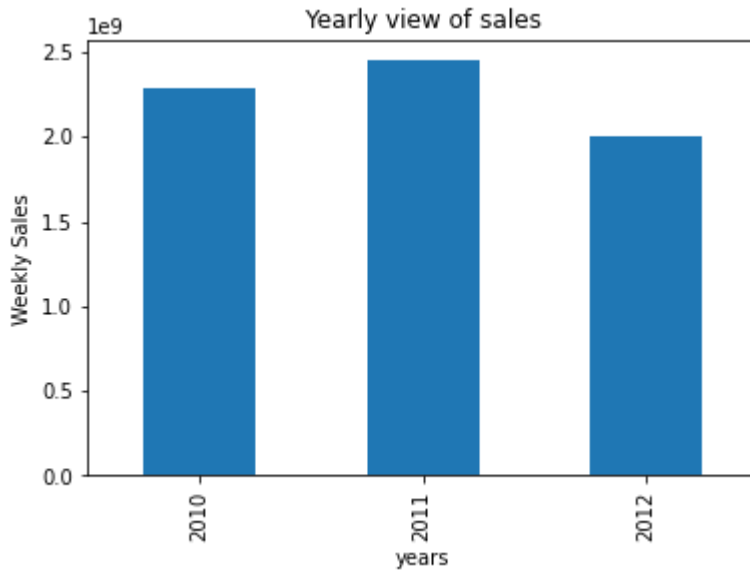
Text(0.5, 1.0, 'Monthly view of sales')



In [60]:

```
# Yearly view of sales
plt.figure(figsize=(10,6))
data.groupby("Year")[["Weekly_Sales"]].sum().plot(kind='bar',legend=False)
plt.xlabel("years")
plt.ylabel("Weekly Sales")
plt.title("Yearly view of sales");
```

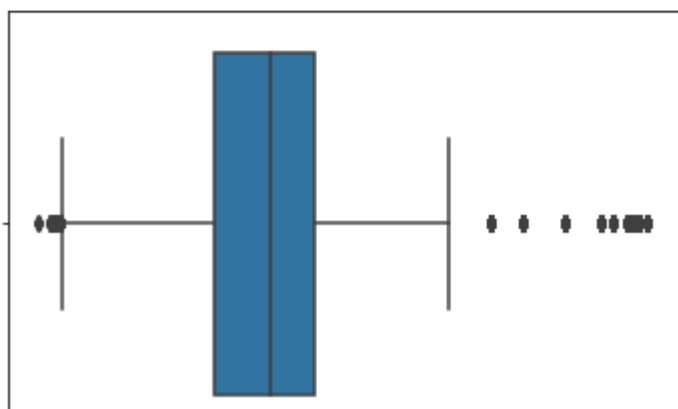
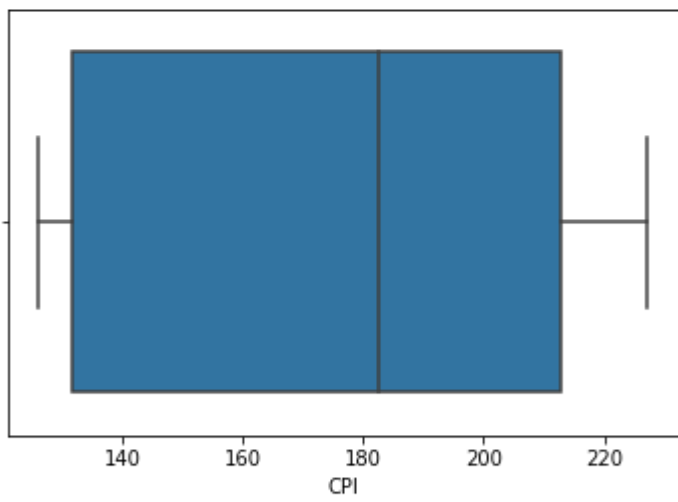
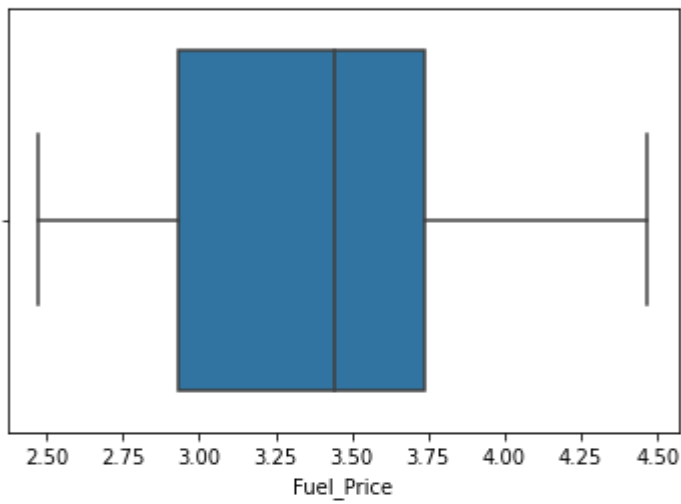
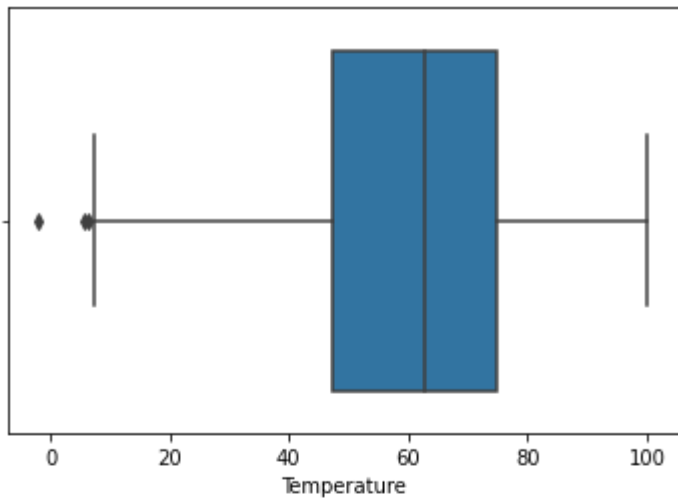
<Figure size 720x432 with 0 Axes>

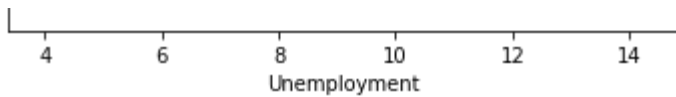


Build prediction models to forecast demand (Modeling)

In [61]:

```
# find outliers
fig, axs = plt.subplots(4,figsize=(6,18))
X = data[['Temperature','Fuel_Price','CPI','Unemployment']]
for i,column in enumerate(X):
    sns.boxplot(data[column], ax=axs[i])
```





In [62]:

```
# drop the outliers
data_new = data[(data['Unemployment']<10) & (data['Unemployment']>4.5) & (data['Temperature']>10)]
data_new
```

Out[62]:

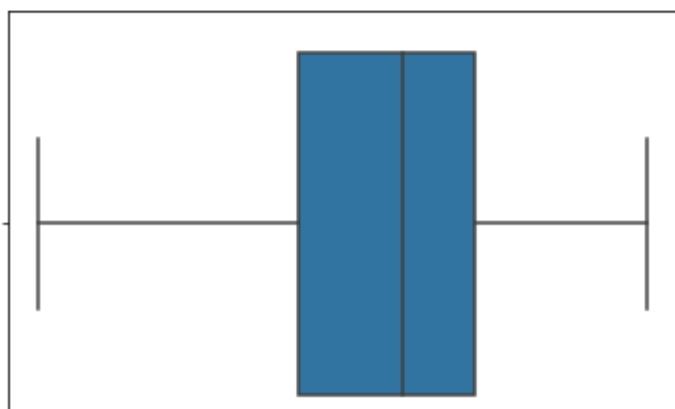
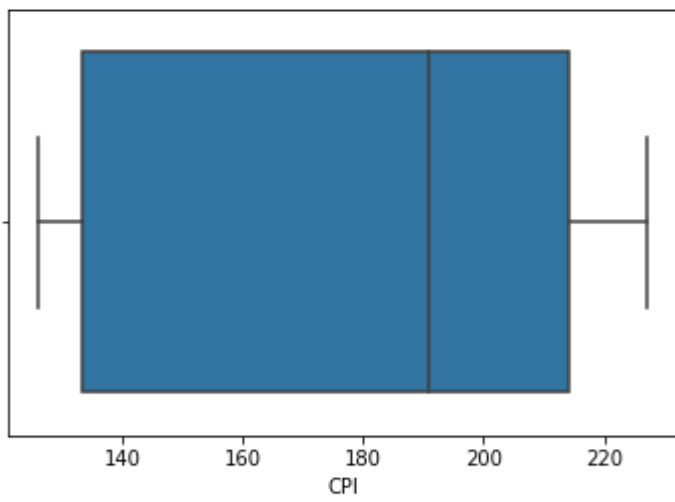
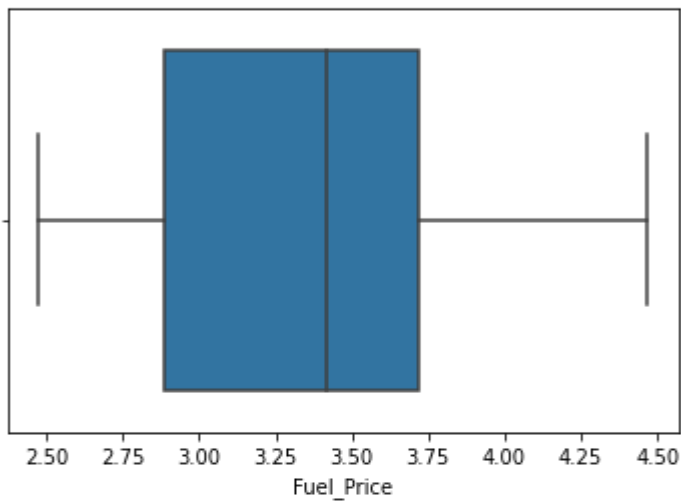
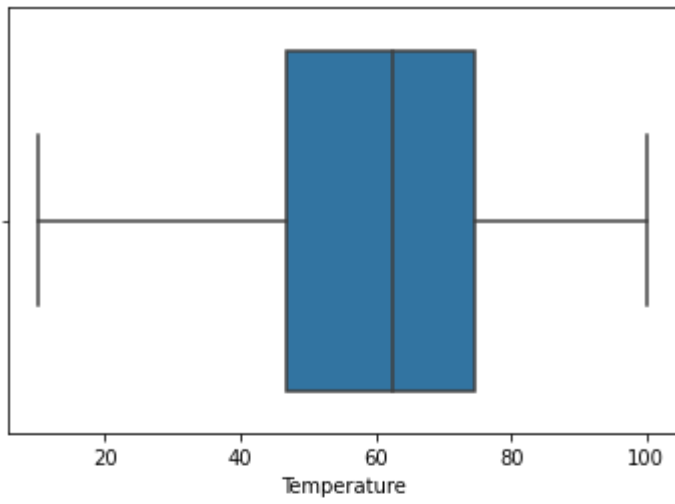
| | Store | Date | Weekly_Sales | Holiday_Flag | Temperature | Fuel_Price | CPI | Unemployr |
|------|-------|------------|--------------|--------------|-------------|------------|------------|-----------|
| 0 | 1 | 2010-05-02 | 1643690.90 | 0 | 42.31 | 2.572 | 211.096358 | 8 |
| 1 | 1 | 2010-12-02 | 1641957.44 | 1 | 38.51 | 2.548 | 211.242170 | 8 |
| 2 | 1 | 2010-02-19 | 1611968.17 | 0 | 39.93 | 2.514 | 211.289143 | 8 |
| 3 | 1 | 2010-02-26 | 1409727.59 | 0 | 46.63 | 2.561 | 211.319643 | 8 |
| 4 | 1 | 2010-05-03 | 1554806.68 | 0 | 46.50 | 2.625 | 211.350143 | 8 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 6430 | 45 | 2012-09-28 | 713173.95 | 0 | 64.88 | 3.997 | 192.013558 | 8 |
| 6431 | 45 | 2012-05-10 | 733455.07 | 0 | 64.89 | 3.985 | 192.170412 | 8 |
| 6432 | 45 | 2012-12-10 | 734464.36 | 0 | 54.47 | 4.000 | 192.327265 | 8 |
| 6433 | 45 | 2012-10-19 | 718125.53 | 0 | 56.47 | 3.969 | 192.330854 | 8 |
| 6434 | 45 | 2012-10-26 | 760281.43 | 0 | 58.85 | 3.882 | 192.308899 | 8 |

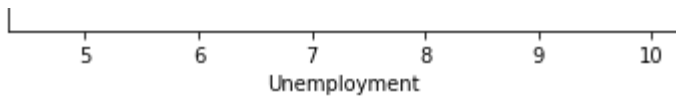
5658 rows × 11 columns



In [63]:

```
# check outliers
fig, axs = plt.subplots(4,figsize=(6,18))
X = data_new[['Temperature', 'Fuel_Price', 'CPI', 'Unemployment']]
for i,column in enumerate(X):
    sns.boxplot(data_new[column], ax=axs[i])
```





Build Model

In [64]:

```
# Import sklearn
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.linear_model import LinearRegression
```

In [65]:

```
# Select features and target
X = data_new[['Store', 'Fuel_Price', 'CPI', 'Unemployment', 'Day', 'Month', 'Year']]
y = data_new['Weekly_Sales']

# Split data to train and test (0.80:0.20)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [66]:

```
# Linear Regression model
print('Linear Regression:')
print()
reg = LinearRegression()
reg.fit(X_train, y_train)
y_pred = reg.predict(X_test)
print('Accuracy:', reg.score(X_train, y_train)*100)

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

sns.scatterplot(y_pred, y_test);
```

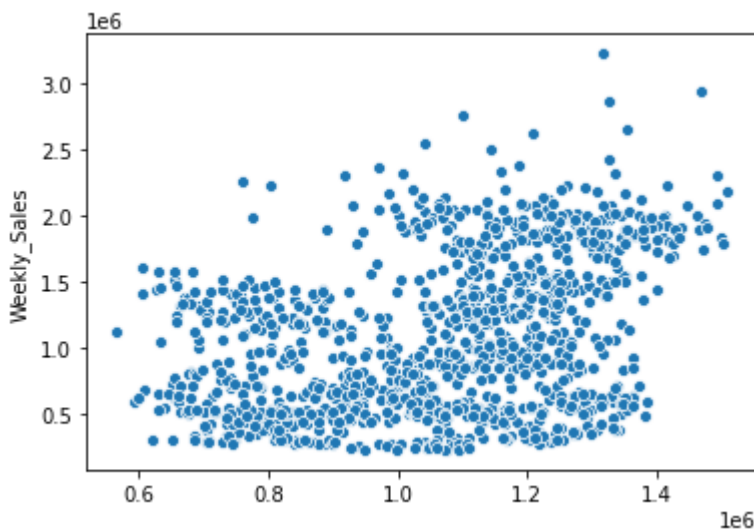
Linear Regression:

Accuracy: 12.512726429399478

Mean Absolute Error: 445859.2353432851

Mean Squared Error: 279505769725.82086

Root Mean Squared Error: 528683.0522400173



```
# Random Forest Regressor
print('Random Forest Regressor:')
print()
rfr = RandomForestRegressor(n_estimators = 400, max_depth=15, n_jobs=5)
rfr.fit(X_train, y_train)
y_pred = rfr.predict(X_test)
print('Accuracy:', rfr.score(X_test, y_test)*100)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
sns.scatterplot(y_pred, y_test);
```

In []:

In []:

In []: